Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital
IMD0041 – Introdução a Organização e Arquitetura de Computares

# Memória Virtual

Prof. Gustavo Girão

# Plano de Aula

- Corrigir exercícios
- Falar sobre a atividade para entrega via sigaa
- Introduzir conceitos básicos para o entendimento do assunto
- Explicar o conceito de memória virtual
- Detalhar o funcionamento da memória virtual

# Introdução

- Uma aplicação, quando executada, deve estar na memória principal
  - Se existirem várias aplicações executando, todas devem estar na memória principal
- O SO também deve estar na memória principal
- Aplicação: instruções + dados

# COMO GERENCIAR A MEMÓRIA PRINCIPAL DE FORMA A COLOCAR TODAS AS APLICAÇÕES E SO NA MEMÓRIA?

No Windows: 440 MBytes


2013 - No Windows: 3 GBytes


No Windows: 4 MBytes

# Princípio I
# a memória precisa ser particionada

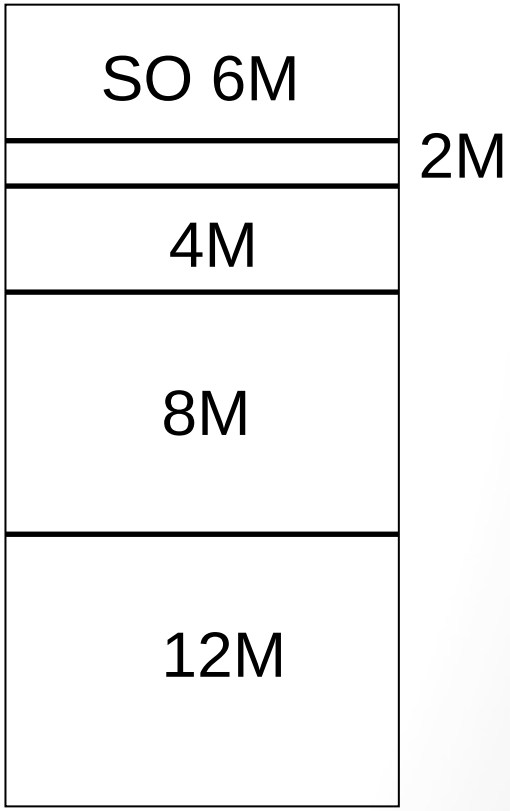
# Particionamento

- Dividir a memória principal em partições
- Cada partição irá armazenar parte do conteúdo de uma aplicação ou SO

# Particionamento

- **Partições de tamanho fixo**
  - Todas as partições têm tamanho fixo, mas não precisam ser do mesmo tamanho


Partições de mesmo tamanho



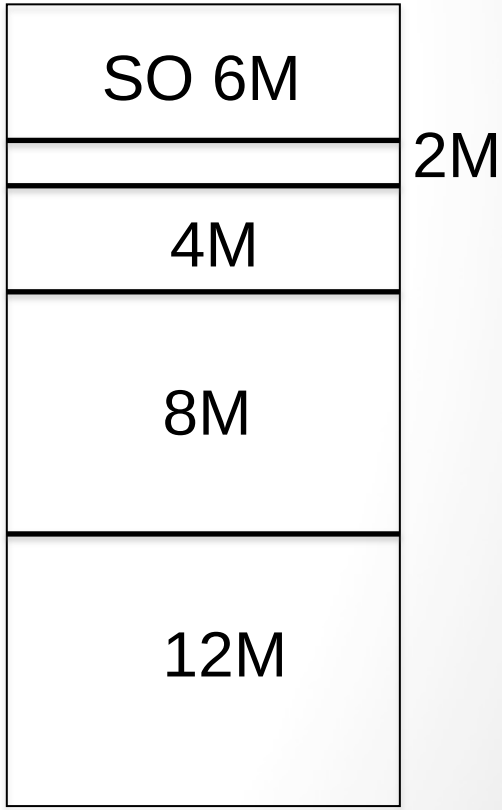

Partições de tamanho variado

# Particionamento

- **Partições de tamanho fixo**
  - Todas as partições têm tamanho fixo, mas não precisam ser do mesmo tamanho

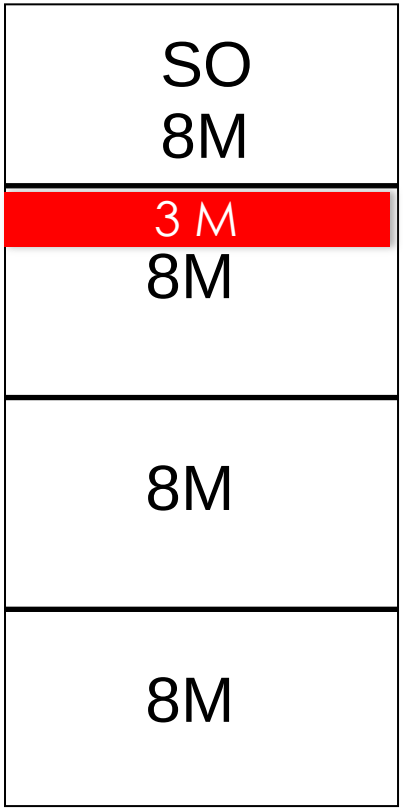

Na maior parte dos casos, a parte da aplicação não exigirá tanta memória quanto a fornecida pela partição

Ex.: 3MB será colocado em 4MB



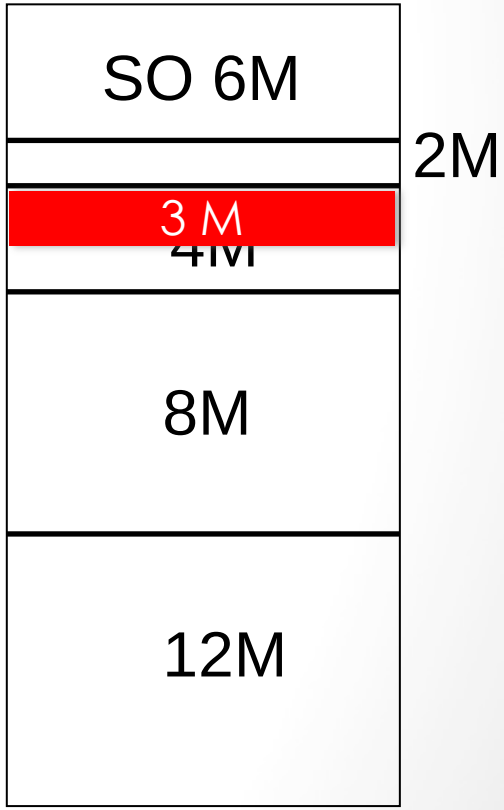
# Particionamento

- **Partições de tamanho fixo**
  - Todas as partições têm tamanho fixo, mas não precisam ser do mesmo tamanho. Pode acarretar em Fragmentação Interna



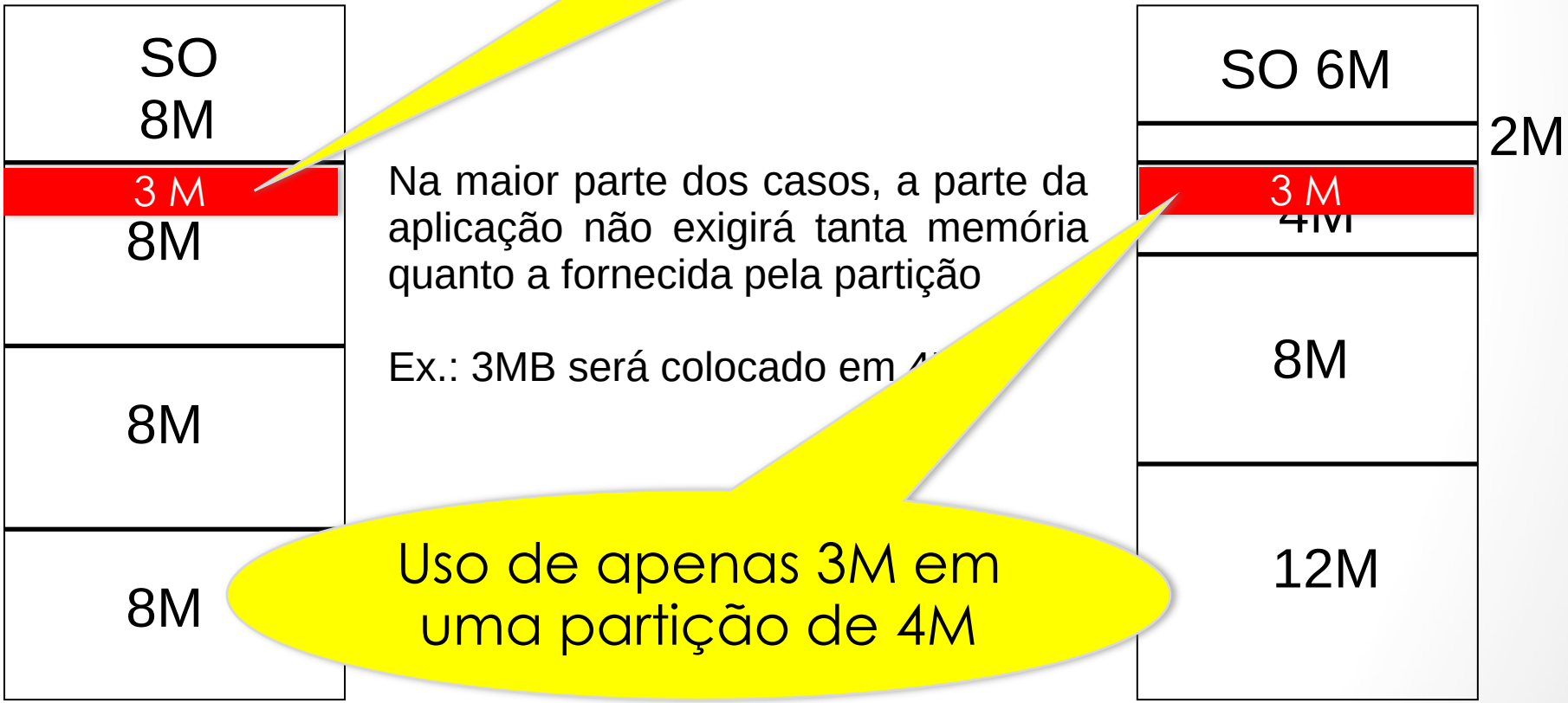

Na maior parte dos casos, a parte da aplicação não exigirá tanta memória quanto a fornecida pela partição

Ex.: 3MB será colocado em 4MB



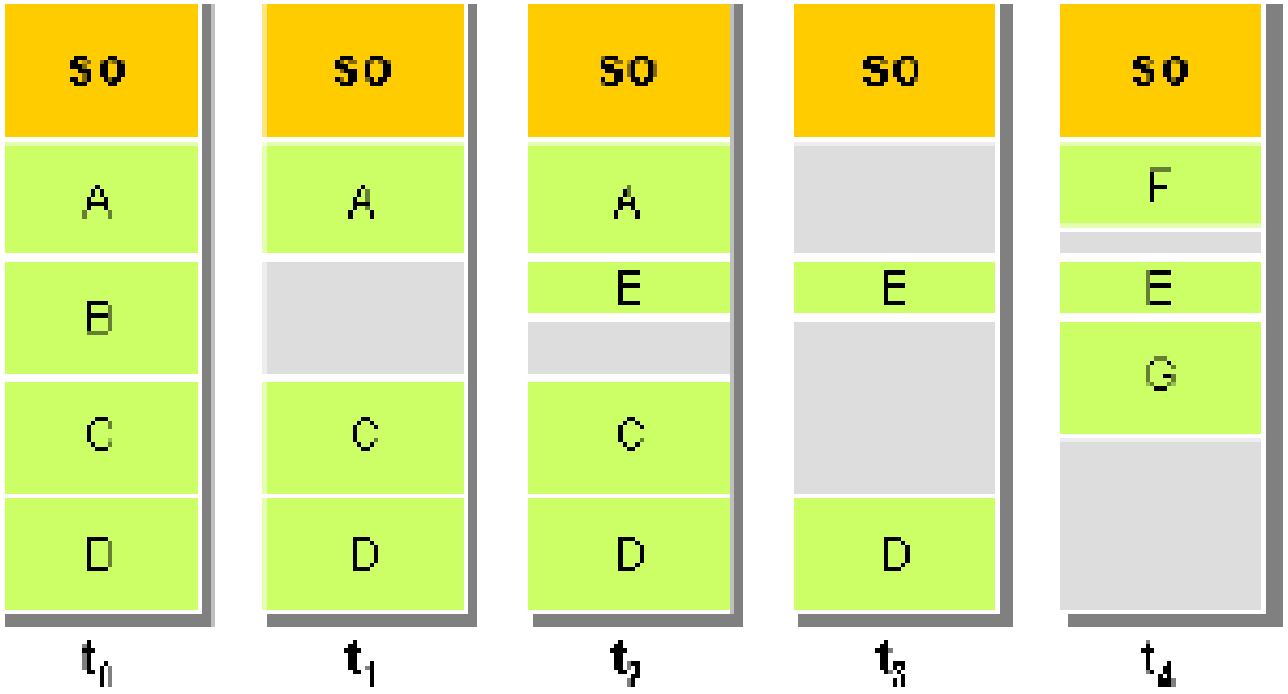
# Particionamento

- **Partições de tamanho**
  - Todas as partições têm tam[illegible]nho, mas não precisam ser do mesm[illegible]anho



# Particionamento

- **Partições de tamanho variável**
  - Quando parte de uma aplicação é levada para a memória, ela recebe somente a quantidade de memória exigida. Pode acarretar em Fragmentação Externa




Fonte: http://www.soffner.com.br/Semana11_SOII.htm

# Particionamento

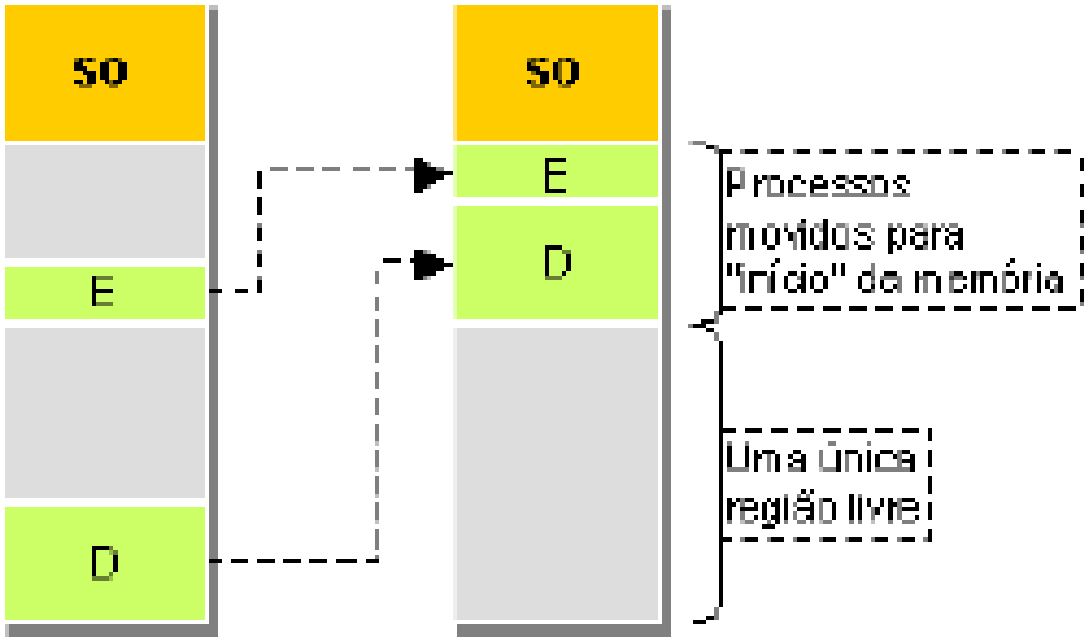
- **Partições de tamanho variável**
  - Quando parte de uma aplicação é levada para a memória, ela recebe somente a quantidade de memória exigida


Fonte: http://www.soffner.com.br/Semana11_SOII.htm

# Particionamento

- **Partições de tamanho variável**
  - Quando parte de uma aplicação é levada para a memória, ela recebe somente a quantidade de memória exigida

Uso de compactação




Fonte: http://www.soffner.com.br/Semana11_SOII.htm

# Particionamento

- Não é possível garantir que um mesmo programa sempre estará na mesma posição da memória sempre que for executado

- Assim, não dá para usar o endereço físico da memória como referência ao programa, pois, em um outro momento, o programa pode estar em outro endereço

# COMO ENCONTRAR OS ENDEREÇOS SE A APLICAÇÃO NÃO ESTÁ EM UMA POSIÇÃO FIXA DA MEMÓRIA?

# Princípio II
# USO DE ENDEREÇOS LÓGICOS PARA REFERENCIAR A MEMÓRIA

# Endereço Lógico e Físico

- **Endereço lógico**
  - Local relativo ao início do programa
- **Endereço físico**
  - Local real na memória principal
- **Existe uma conversão de endereço lógico em um físico.**

# Princípio III
# PARTICIONAR A MEMÓRIA DE MODO A EVITAR O DESPERDÍCIO

# Paginação

- Partições desiguais de tamanho fixo e de tamanho variável são ineficazes

- Solução:
  - Dividir a memória em pedaços menores de tamanho fixo – **Frames**
  - Dividir a aplicação em pedaços de mesmo tamanho dos pedaços da memória – **Páginas**
  - No máximo, o espaço desperdiçado na memória para esse processo é uma fração da última página

# Paginação

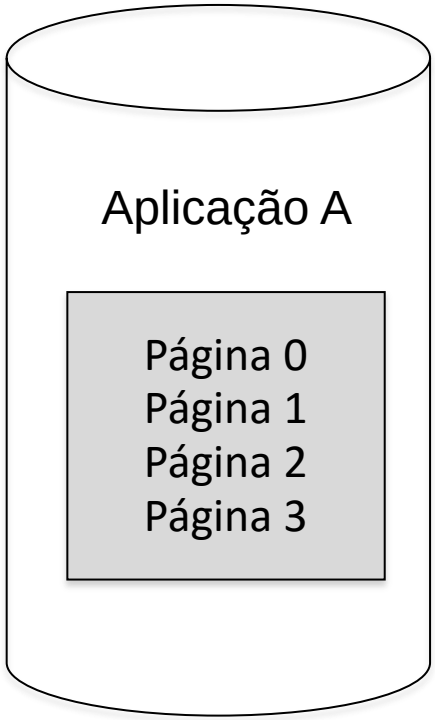
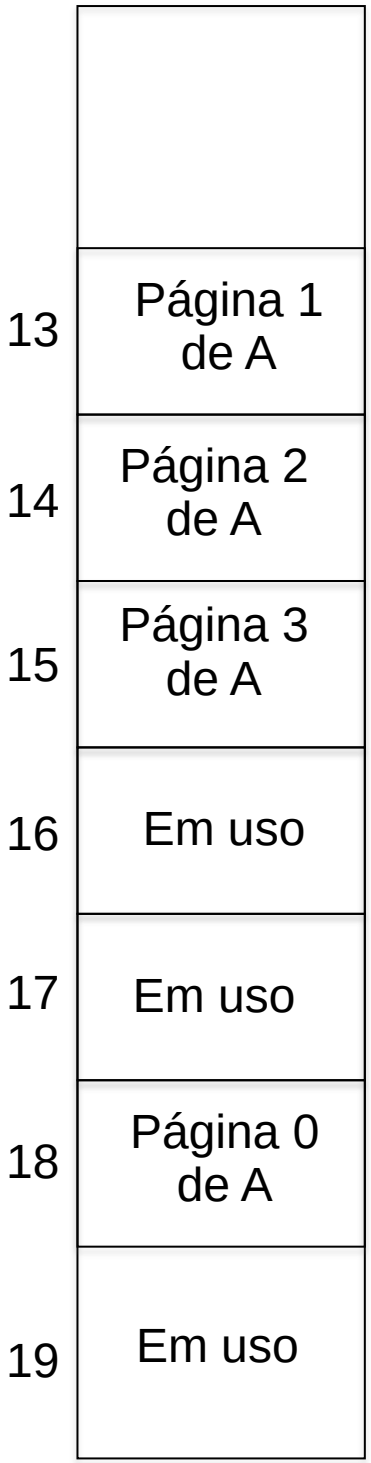



Tabela de página da Aplicação A

| |
|---|
| 18 |
| 13 |
| 14 |
| 15 |



# Paginação por Demanda

- Cada página de uma aplicação é trazida apenas quando necessária, ou seja, por demanda
  - **Princípio da localidade**

# Paginação por Demanda

- Se precisar de outra página: FALTA DE PÁGINA (*page fault*)
  - Busca no disco
- Possível mapear mais aplicações na memória
- Páginas não usadas não precisam ser mantidas na memória
- Uso de política de substituição de página
- Alta taxa de falta de página: ***thrashing***

# Paginação por Demanda

- Se precisar de outra página: FALTA DE PÁGINA (*page fault*)
  - Bus
- Possí
  mem
- Págin
  mantidas na memória
- Política de substituição de página
- Alta taxa de falta de página: ***thrashing***

ONDE JÁ VIMOS ISSO?

# MEMÓRIA VIRTUAL

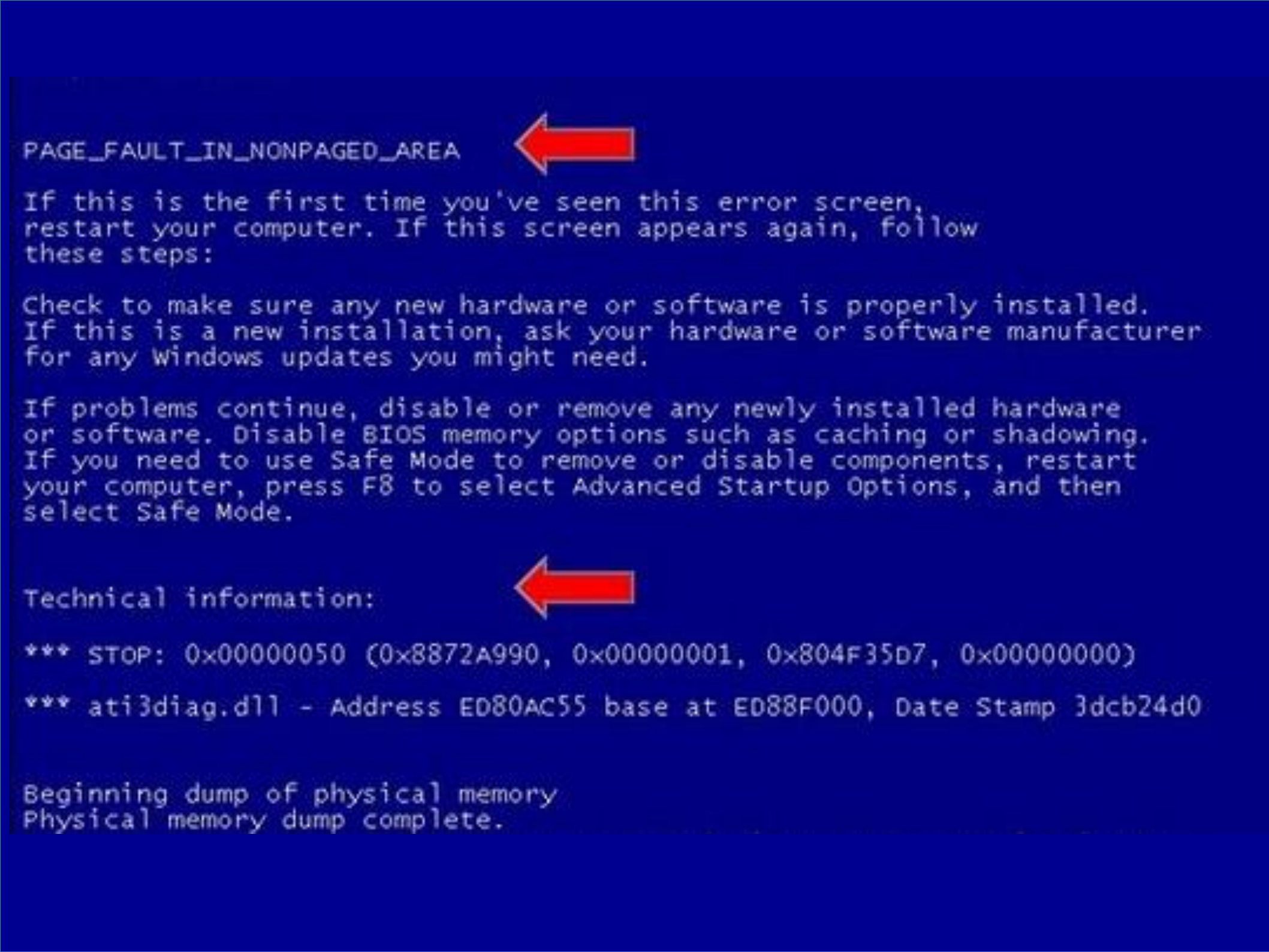
# Memória Virtual

- **Princípio**: Não é necessário carregar todo o programa na memória
- Se o programa for maior que a memória, é necessário criar maneiras de estruturá-lo em partes que possam ser carregadas uma de cada vez
  - Com a paginação por demanda, quem faz isso é SO e hardware
  - Para o programador existe uma memória imensa (a troca de páginas é transparente)

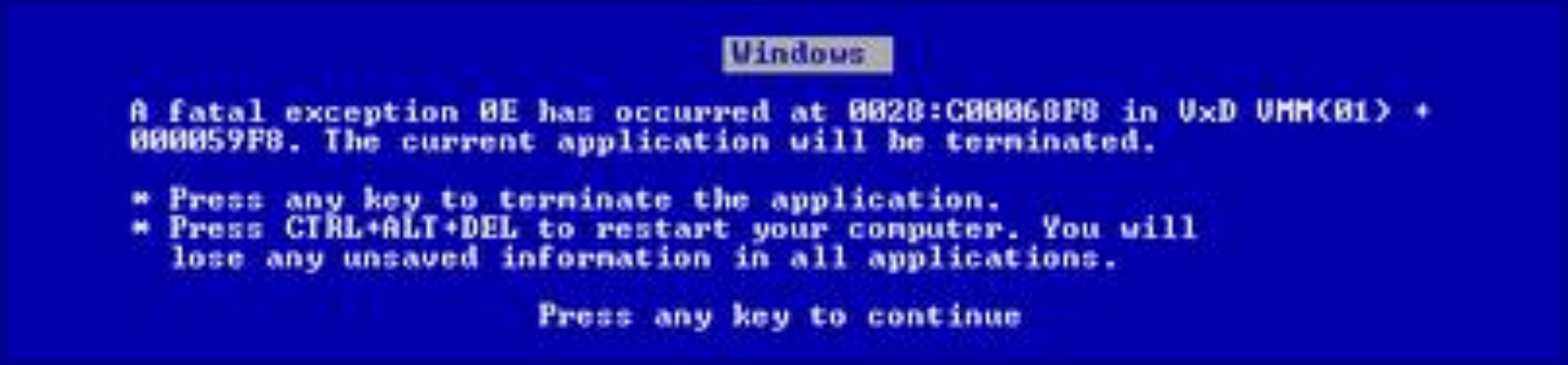
# Memória Virtual

- **Princípio**: Não é necessário carregar todo o programa na memória
- S
  n
  o
  - 
  - 

O QUE ACONTECERIA SE NÃO HOUVESSE A TROCA DE PÁGINA E A MEMÓRIA PRINCIPAL ESTIVESSE CHEIA?


PAGE_FAULT_IN_NONPAGED_AREA
If this is the first time you've seen this error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x00000050 (0x8872A990, 0x00000001, 0x804F35D7, 0x00000000)
*** ati3diag.dll - Address ED80AC55 base at ED88F000, Date Stamp 3dcb24d0
Beginning dump of physical memory
Physical memory dump complete.

Desculpe, mas você não pode carregar mais nenhum aplicativo. Por favor, feche um dos programas abertos para poder abrir um novo



# Memória Virtual

- **Funcionamento**
- **Desempenho**
- **Papel na hierarquia de memória**

# Introdução

- Memória Virtual é a técnica que dá ao programador a ilusão de poder acessar rapidamente um grande espaço de endereçamento.
- Objetivos da técnica:
  - Permitir que haja um meio seguro e eficiente de se compartilhar informações, armazenadas na memória, entre vários programas
  - Minimizar os problemas causados aos programas pela existência de uma pequena quantidade de memória principal

# Introdução

- Os programas que compartilham a memória de determinada máquina mudam dinamicamente durante o processo de execução.

- Cada programa deve ser compilado usando seu próprio espaço de endereçamento (ou seja, em uma região da memória acessível somente a esse programa).

- **A técnica de memória virtual realiza a tradução do espaço de endereçamento de um programa para seus endereços reais.**

# Introdução

- A técnica de memória virtual permite que o tamanho de um único programa exceda a quantidade total de memória real disponível para sua execução

- A técnica de memória virtual gerencia automaticamente dois níveis de hierarquia:
  - Memória principal
  - Memória secundária ou auxiliar

# Introdução

- **A memória virtual faz com que a memória principal funcione como uma cache para memória secundária (discos magnéticos)**

- Vantagens:
  - Ilusão de ter mais memória física
  - Realocação de programa
  - Proteção entre processos
  - Separação entre memória lógica e memória física

# Memória Virtual x Memória Cache

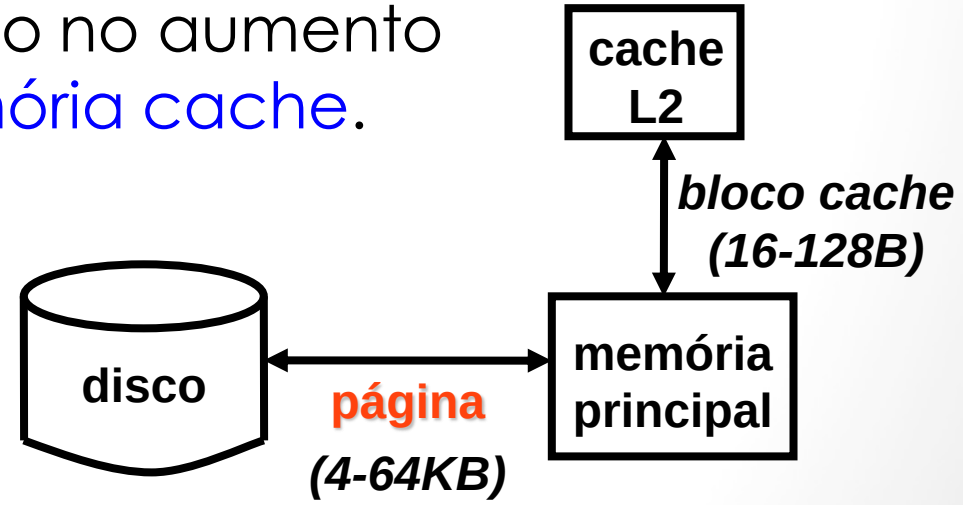
| Memória Virtual | Memória cache |
| --- | --- |
| Página ou Segmento | Bloco |
| Falta de página | Miss (falha) |
| Substituição de erros controlado pelo Sistema Operacional | Substituição de erro controlada pelo hardware |

# Memória Virtual x Memória Cache

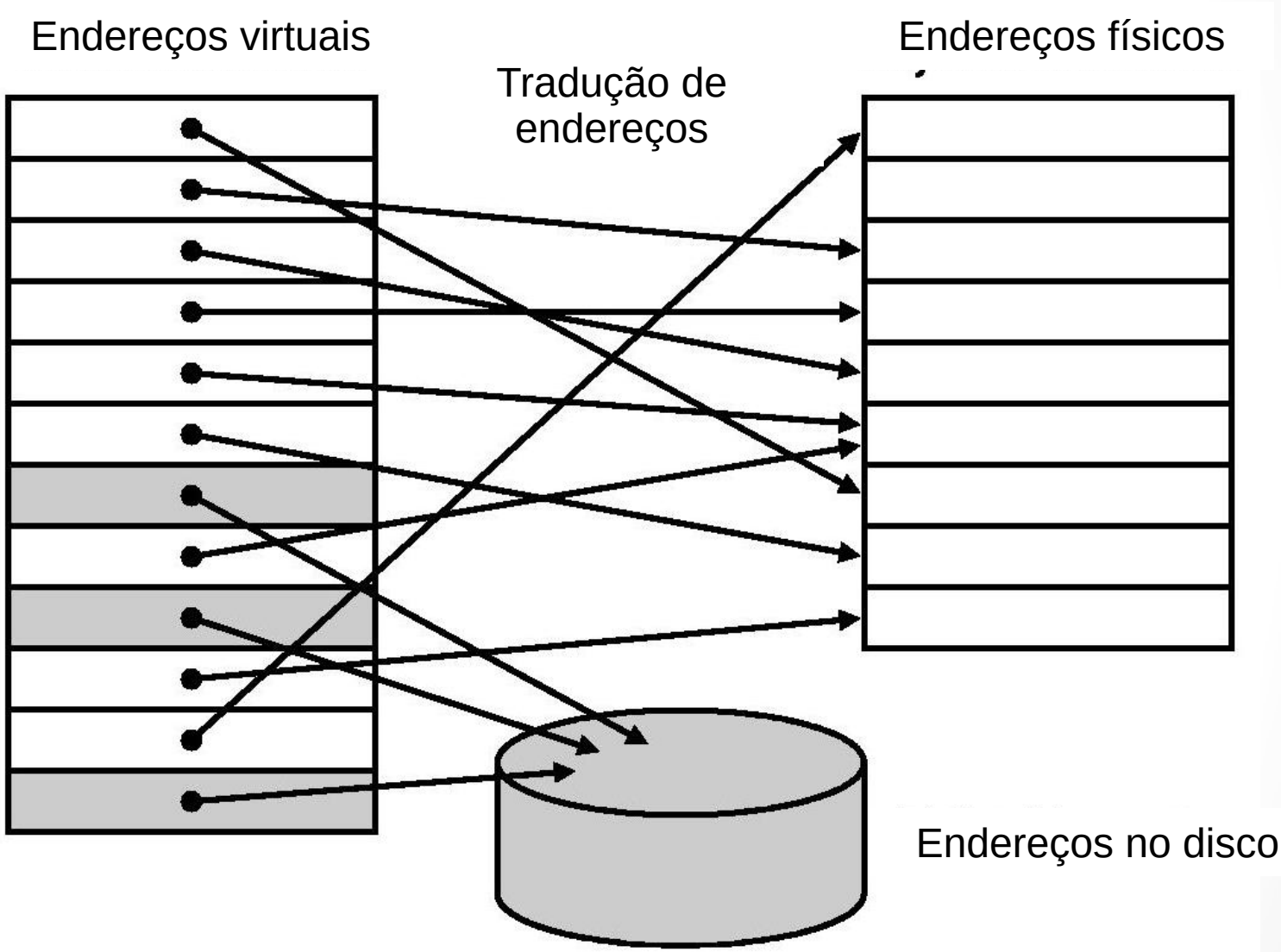
| Parâmetro | Cache L1 | Memória virtual |
|---|---|---|
| Tamanho do bloco (página) | 16-128 bytes | 4.096-65.536 bytes |
| Tempo de acerto (hit) | 1-3 ciclos | 100-200 ciclos |
| Penalidade de falha | 8-200 ciclos | 1 – 10 M ciclos |
| Taxa de falha | 0,1-10% | 0,00001-0,001% |

# Funcionamento

- Transferência entre disco e memória ocorre em páginas, cujo tamanho é definido pelo ISA (conjunto de instruções).
- Tamanho da página é grande para amortizar o alto custo de acesso ao disco.
- Compromisso (*tradeoff*) no aumento do tamanho das páginas na memória virtual é similar ao compromisso no aumento dos blocos para a memória cache.



# Memória Virtual – tradução de endereços



# Memória Virtual – tradução de endereços

- Tradução de endereços é feita pelo hardware e pelo sistema operacional (SO).
- SO mantém para cada programa:
  - Quais páginas estão associadas a ele;
  - Onde fica cada página no disco;
  - Quais páginas estão residentes na memória;
  - O nº de cada página física associada com o nº da página virtual residente na memória.

I. Posicionamento da página.
Onde a página deve ser colocada na memória principal?

II. Identificação da página.
Como a página é encontrada na memória principal?

III. Substituição de página.
Quais páginas serão trocadas em uma falta?

IV. Estratégia de gravação.
O que acontece em uma escrita de página?

# I – Posicionamento da Página

- A penalidade de erro para a memória virtual é muito alta, pois envolve o acesso a um dispositivo de armazenamento magnético rotativo.

- Em razão disso, para reduzir a frequência de faltas de páginas, os sistemas operacionais utilizam o esquema de posicionamento totalmente associativo.

# II - Identificação da Página

- A desvantagem da escolha do posicionamento totalmente associativo está em localizar uma entrada, já que ela pode estar em qualquer lugar da memória virtual.

- Como o espaço ocupado pela memória virtual é maior que aquele ocupado pela memória cache, o tempo de busca aumenta substancialmente.

- Para contornar essa desvantagem, utiliza-se uma tabela de páginas, uma estrutura que indexa as traduções de endereços virtuais para endereços físicos.

# II - Identificação da Página

- Tabela de páginas está armazenada na memória principal

- É indexada com o número da página extraído do endereço virtual e contém o número da página física correspondente

# II - Identificação da Página

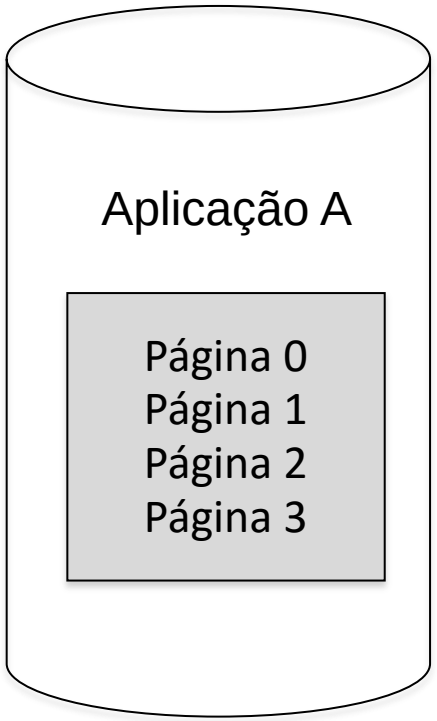
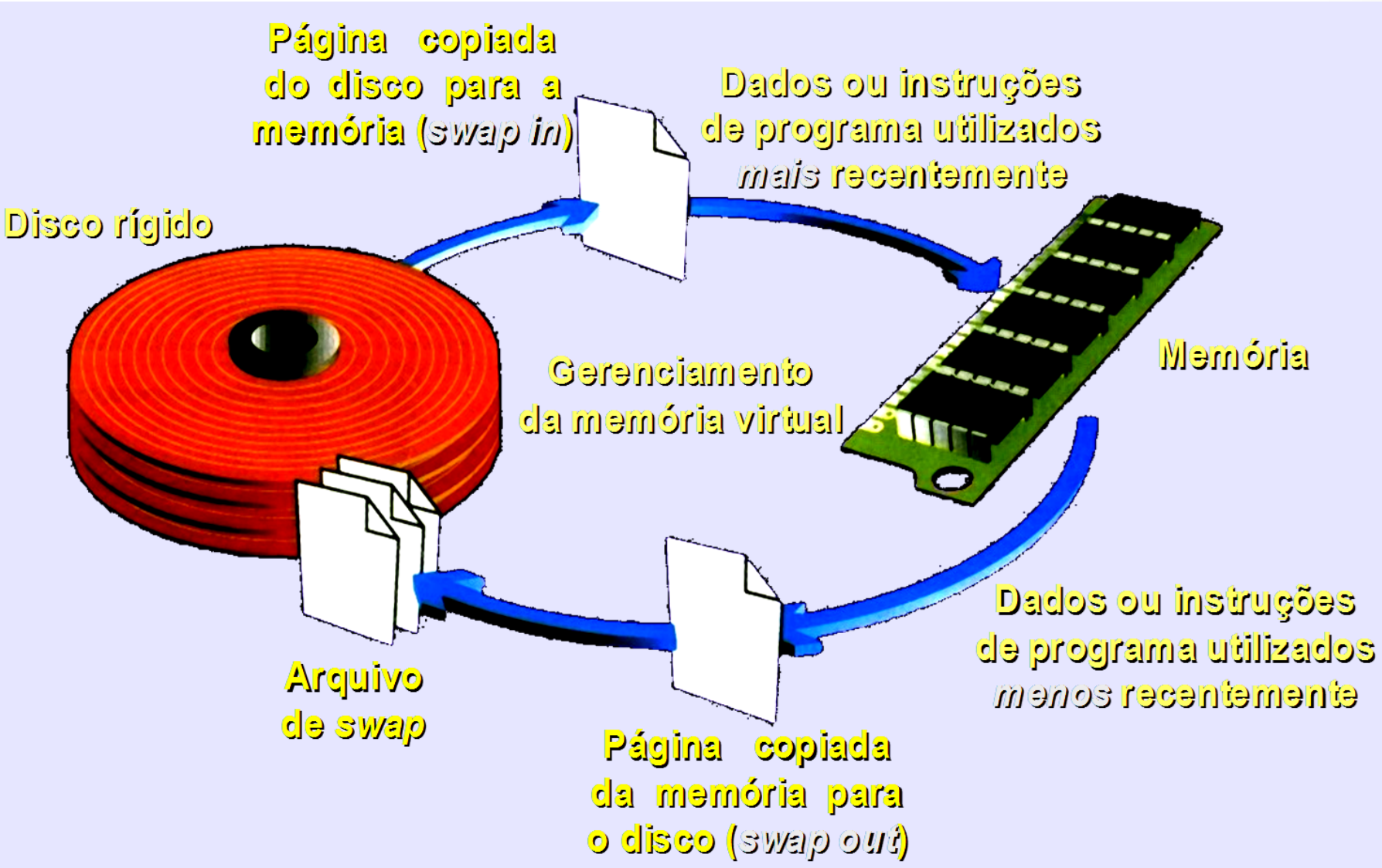



Tabela de página da Aplicação A

| |
|---|
| 18 |
| 13 |
| 14 |
| 15 |

| | |
|---|---|
| | |
| 13 | Página 1 de A |
| 14 | Página 2 de A |
| 15 | Página 3 de A |
| 16 | Em uso |
| 17 | Em uso |
| 18 | Página 0 de A |
| 19 | Em uso |

# III – Substituição da Página



# III – Substituição da Página

- Quando ocorre falta de página:
  - O sistema operacional precisa
    - Encontrar a página faltante no nível hierárquico inferior (geralmente, no HD)
    - Decidir em que lugar da memória principal deve ser colocada a página requisitada
  - O endereço virtual não informa em que posição do HD está a página que gerou a falta de página.

# III – Substituição da Página

- Em uma memória totalmente associativa, todos os blocos são candidatos à substituição.

Estratégias para a substituição de blocos:

**Aleatória**: os blocos candidatos à substituição são escolhidos ao acaso, possivelmente contando com algum auxílio de hardware.

Bloco menos usado recentemente (**LRU**): o bloco substituído é aquele menos utilizado recentemente.

# IV – Estratégia de Gravação

- A escrita no disco consome de 1 milhão a 10 milhões de ciclos de clock.
- Esquema write-through não funciona para memória virtual.
- Esquema write-back é usado:
  - Página é copiada para o disco no momento em que for substituída (nomenclatura: "copy-back")

# OUTRA FORMA DE DIVIDIR A MEMÓRIA

# Segmentação

- Paginação é um sistema de memória virtual unidimensional
  - Os endereços variam de 0 a um endereço máximo

- Em alguns casos, é mais interessante a existência de dois ou mais espaços de endereçamento virtual separados para um mesmo programa

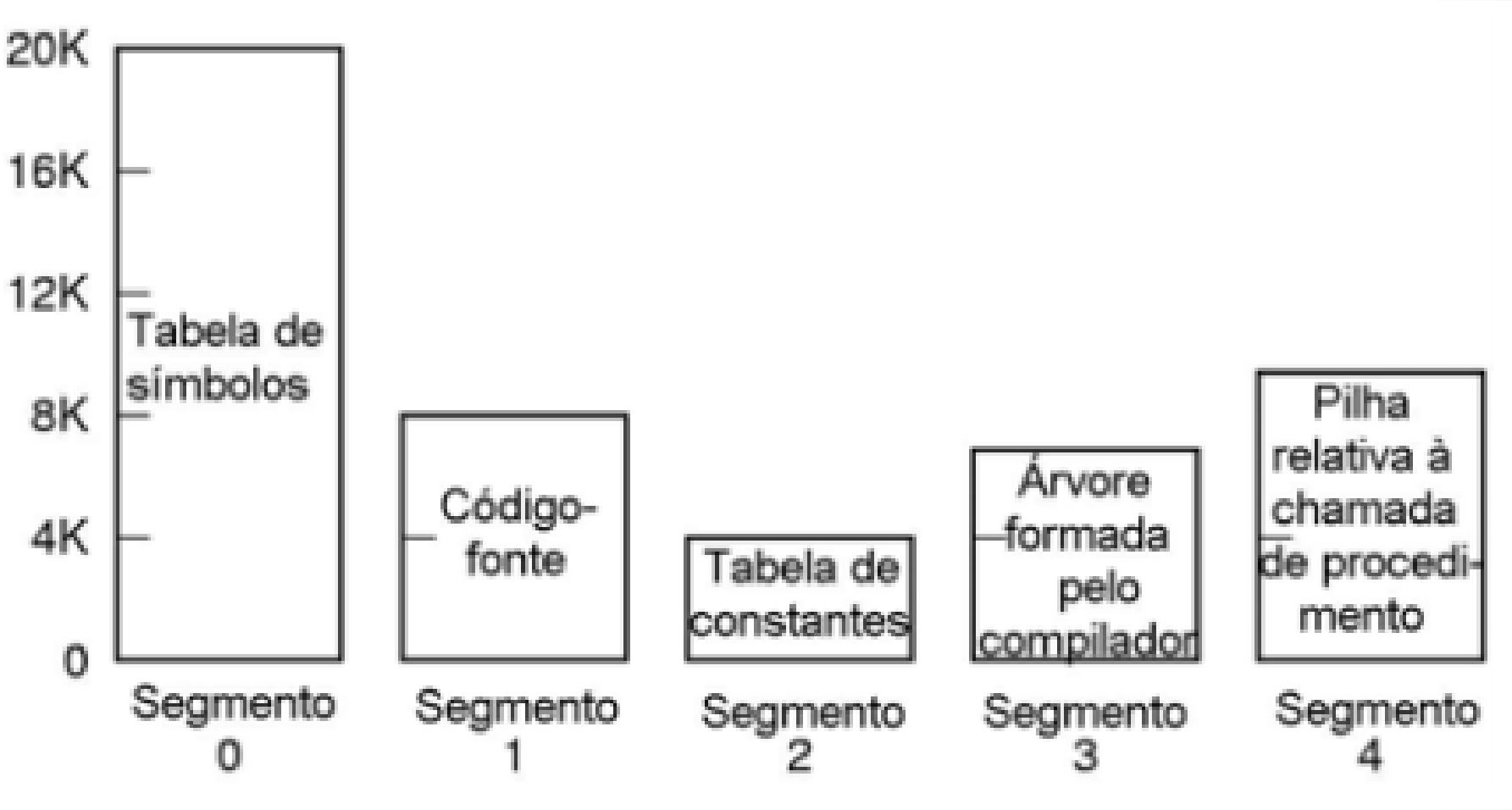
# Segmentação

- Segmentação: vários espaços de endereçamento completamente independentes
  - Esses espaços são chamados de segmentos
  - Cada segmento é composto de uma sequência de endereços, de 0 até um valor máximo
  - O tamanho de cada segmento não precisa ser fixo
    - Segmentos diferentes podem ter tamanhos diferentes

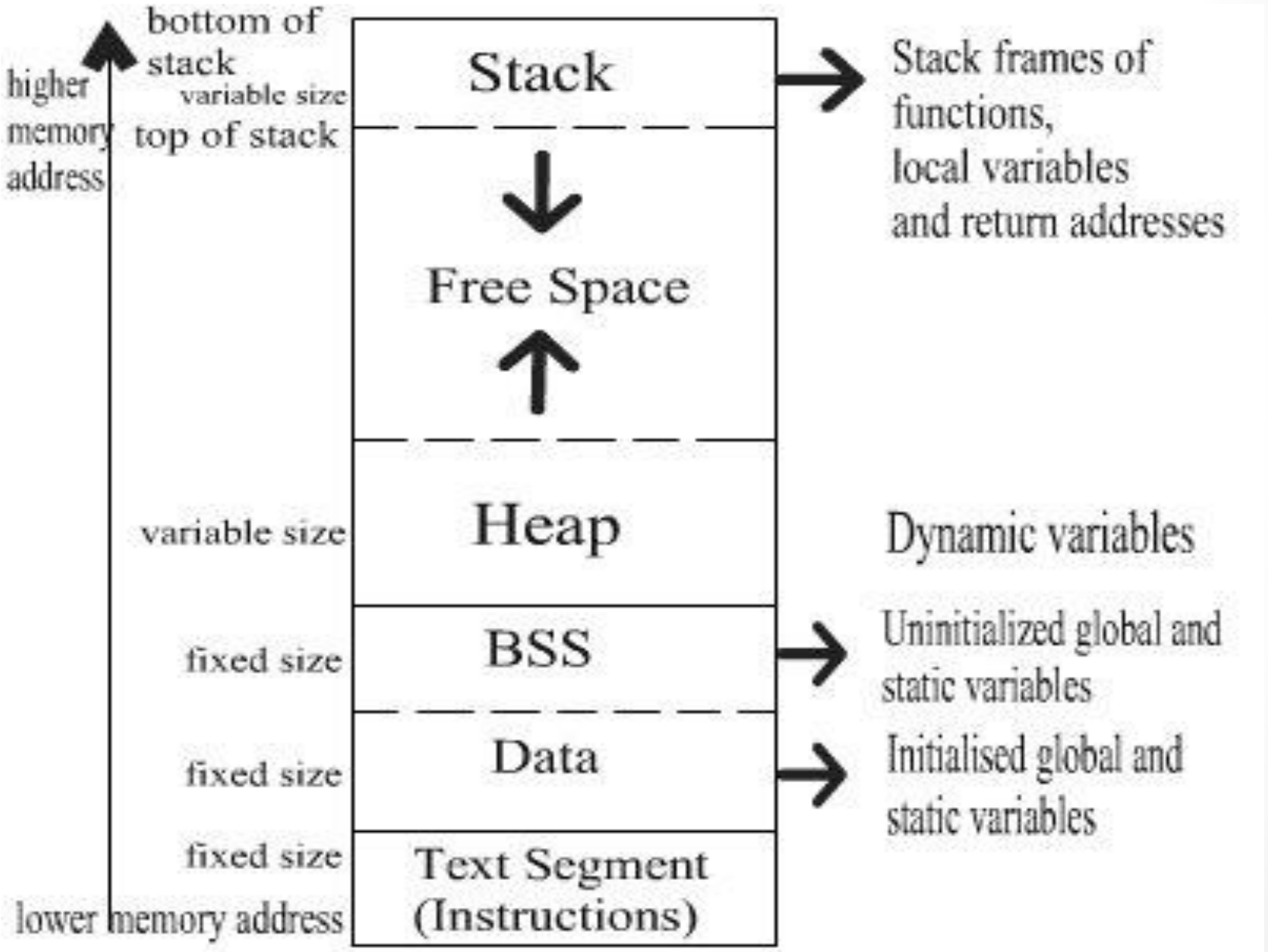
# Segmentação

- Cada segmento constitui um espaço de endereçamento separado
  - Aumentar ou diminuir um segmento não influencia nos outros
- O endereço é especificado em duas partes
  - Número do segmento
  - Endereço do segmento

# Segmentação



# Segmentação – Programa em C

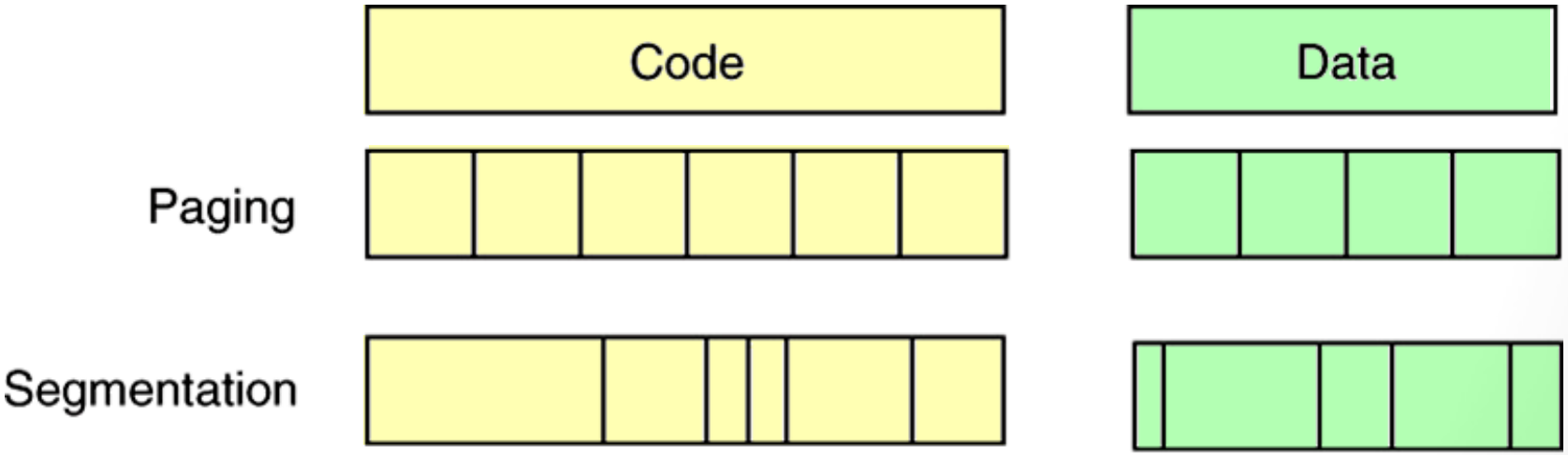


# Segmentação

- Vantagem
  - Simplifica o tratamento de estruturas de dados que crescem (a ED recebe seu próprio segmento)
  - Permite que os programas sejam alterados e recompilados de modo independente
  - Compartilhamento entre processos
  - Proteção entre programas

# Página X Segmento

- A paginação utiliza blocos de tamanho fixo.
- A segmentação utiliza blocos de tamanho variável.
- Um segmento consiste de duas partes:
  - Número de segmento
  - Offset de segmento



# Página X Segmento

- Por que utilizar segmentação?
  - Suporte a métodos de proteção mais avançados
    - Segmento de código: só pode ser executado
    - Segmento de dado: pode ser lido ou escrito mas não executado
  - Compartilhar um espaço de endereçamento

- Desvantagem:
  - Divide o espaço de endereço em partes logicamente separadas, que precisam ser manipuladas como um endereço de duas partes
  - Necessidade de compactação

# Página X Segmento

- A decisão de usar memória virtual paginada ou memória virtual segmentada afeta o desempenho da CPU

| | Página | Segmento |
|---|---|---|
| Palavras por endereço | Uma | Duas (segmento e offset) |
| Visível ao programador? | Invisível ao programador de aplicação | Pode ser visível ao programador de aplicação |
| Substituição de blocos | Trivial (todos os blocos são do mesmo tamanho) | Difícil (deve encontrar porções contínuas e sem uso da memória principal) |
| Eficiência de uso de memória | Fragmentação interna (porção não usada da página) | Fragmentação externa (porções não usadas na memória principal) |
| Eficiência de tráfego de disco | Sim (ajuste do tamanho de página para balancear tempo de acesso e de transferência) | Nem sempre (pequenos segmentos podem transferir poucos bytes apenas) |

# Página X Segmento

- A decisão de usar memória virtual paginada ou memória virtual segmentada afeta o desempenho da CPU

| | Página | Segmento |
|---|---|---|
| Palavras por endereço | Uma | Duas (segmento e offset) |
| Visível ao programador? | Invisível ao programador de aplicação | Pode ser visível ao programador de aplicação |
| Substituição de blocos | Trivial (todos os blocos são do mesmo tamanho) | Difícil (deve encontrar porções contínuas e sem uso da memória principal) |
| Eficiência de uso de memória | Fragmentação interna (porção não usada da página) | Fragmentação externa (porções não usadas na memória principal) |
| Eficiência de tráfego de disco | Sim (ajuste do tamanho de página para balancear tempo de acesso e de transferência) | Nem sempre (pequenos segmentos podem transferir poucos bytes apenas) |

# Proteção com memória virtual

# Proteção com Memória Virtual

- Função de um sistema de memória virtual:

> Permitir que uma única memória principal seja compartilhada por vários processos, oferecendo proteção de memória entre eles e o SO.

- Mecanismo de proteção precisa garantir que:
  - Um processo "rebelde" não possa escrever no espaço de endereçamento de outro processo de usuário ou no espaço do SO.
  - Um determinado processo não leia dados de outro processo.

# Proteção com Memória Virtual

- Um processo deve operar corretamente, esteja em:
  - Execução contínua, ou
  - Interrupção repetida e alternante com outros processos.

- A responsabilidade pelo comportamento correto dos processos é compartilhada entre:
  - Arquitetura – deve assegurar que o estado da CPU seja salvo e restaurado, ao rodar um determinado processo.
  - Sistema Operacional – processos não podem interferir uns com os outros.

# Proteção com Memória Virtual

- Cada processo tem, ao mesmo tempo, na memória principal:
  - Espaço de endereçamento virtual
  - Estado da CPU
  - Espaço de instruções
  - Espaço de dados

- S.O. deve manter as tabelas de páginas organizadas de modo que:
  - Páginas virtuais independentes sejam mapeadas para páginas físicas disjuntas.
  - Assim, um processo não será capaz de acessar as informações de outro.

# Proteção com Memória Virtual

Um processo de usuário não é capaz de modificar o mapeamento da tabela de páginas.

O SO proíbe o processo de usuário de modificar sua própria tabela de páginas. Mas o SO precisa poder modificar qualquer das tabelas de páginas.

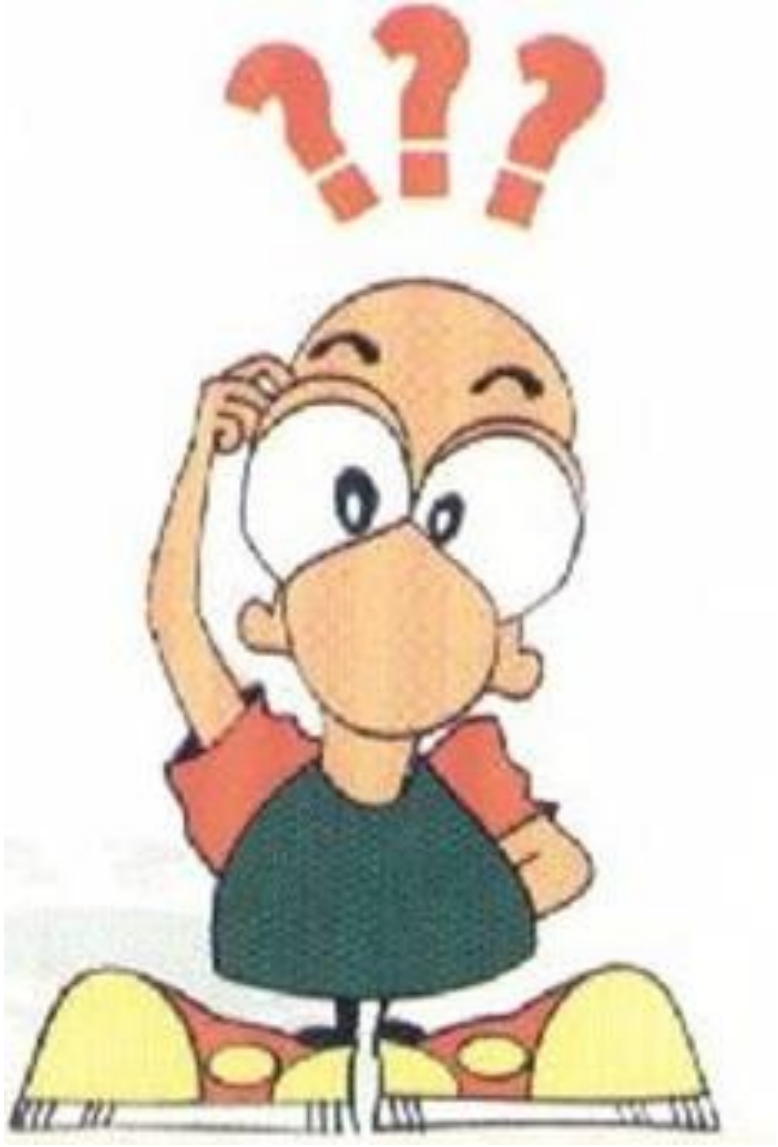
Por isso, as tabelas de páginas são colocadas no espaço de endereçamento do SO.

# Compartilhando informações entre processos

- Processos podem precisar compartilhar informações entre si para leitura, mas não para escrita.
- Exemplos:
  - Múltiplas instâncias de uma mesmo programa em execução.
  - Múltiplas instâncias de uma mesma biblioteca em uso por vários programas.
- Portanto, um bit de acesso à escrita pode ser incluído na tabela de páginas a fim de restringir o compartilhamento apenas à leitura.
- Como o restante da tabela de páginas, esse bit de acesso à escrita só pode ser modificado pelo SO.

# Compartilhando informações entre processos

- Para permitir que um processo A leia uma página pertencente ao processo B:
  - B solicita ao SO a criação de uma entrada na tabela de páginas para uma página virtual no espaço de endereçamento de A que aponte para uma página física que B deseja compartilhar.
  - SO pode usar o bit de escrita para evitar que A escreva nas informações acessadas, se B assim o desejar.
  - Bits que determinam os direitos de acesso a uma página precisam ser incluídos na tabela de páginas.



# Para saber mais ...

- PATTERSON, D.A. & HENNESSY, J. L. **Organização e Projeto de Computadores** - A Interface Hardware/Software. 3ª ed. Campus, 2005.

- William Stallings, “Arquitetura e Organização de Computadores,” 8ª Edição

  - **Capítulo 8 – (sessão 8.3)**